PROVA FINALE

ING. SOFTWARE

Documentazione del progetto “Lorenzo il Magnifico”.

Daverio Pietro e Marotti Michele

Manuale utente

All'avvio dell'applicazione viene mostrata la maschera di login, nella quale l'utente deve scegliere il proprio username, il metodo di comunicazione con il server (Socket o RMI) e se giocare utilizzando l'interfaccia grafica oppure tramite riga di comando. Dopo aver terminato la fase di accesso, l'utente viene associato a una partita ancora da cominciare.

Graphical User Interface

Inizialmente l'utente ha a disposizione solo i bottoni per cominciare la partita o per abbandonare.

La partita comincia quando un giocatore preme il bottone "Start Game" (ma solo se è presente almeno un altro giocatore in attesa di cominciare a giocare), oppure automaticamente quando viene raggiunta la quota di 4 giocatori in attesa.

L'applicazione si adatta alle dimensioni dello schermo portandosi in modalità full-screen. Nella parte sinistra troviamo l'intero tabellone del gioco, la parte destra invece è divisa in due, sopra si trovano le informazioni degli avversari, mentre sotto si ha la plancia del giocatore con i propri familiari.

All'avvio della partita vengono disposte tutte le pedine sul tabellone di gioco e viene comunicato il giocatore che deve effettuare la prima mossa. Ogni giocatore può vedere le proprie carte e le proprie risorse nella plancia, e può consultare le informazioni degli avversari che vengono sempre mostrate nello spazio dedicato. (immagine a lato)

Prospetto di un giocatore avversario

Per vedere le carte in possesso degli avversari sarà sufficiente posizionarsi con il mouse sopra l'icona del tipo di carta di interesse. I punti militari, fede e vittoria sono sempre ottenibili guardando la posizione delle pedine dei giocatori sul tabellone.

All'inizio di ogni turno vengono posizionate le carte nelle torri e viene comunicato il valore dei dadi. Passando con il mouse sopra le carte nella torre, quest'ultime verranno ingrandite per facilitarne la lettura da parte dell'utente.

Per poter effettuare una mossa, il giocatore deve selezionare il familiare che intende piazzare, e successivamente cliccare sullo spazio azione sul quale desidera piazzarlo. Per tutti gli spazi azione, dopo la selezione, viene mostrata una finestra richiedendo il numero di servitori che si intende aggiungere per aumentare il valore dell'azione. Nel caso di spazi azione della torre, nella finestra viene mostrata anche la carta sviluppo associata. Prima di confermare l'acquisto della carta, può essere necessario che l'utente scelga l'effetto che desidera attivare (per alcune carte è possibile effettuare uno solo degli effetti, o costi, proposti). Se l'acquisto va a buon fine, il giocatore vedrà comparire la carta nella propria plancia e le proprie risorse aggiornate (eventualmente si saranno spostate anche le pedine sul tabellone). Se invece non è possibile portare a termine l'operazione verrà mostrato un messaggio di errore, e l'utente sarà invitato a selezionare un altro spazio azione per completare la propria mossa.

Scelta effetto

Tutte le volte che viene attivato l'effetto relativo ai privilegi del consiglio, comparirà una pop up tramite la quale l'utente potrà scegliere il compenso che gli spetta.

Per le azioni bonus invece, dovute all'effetto di un'altra carta, non sarà necessario scegliere un familiare, ma sarà sufficiente premere sullo spazio azione desiderato. Ogni azione bonus può essere effettuata solo su alcuni spazi azione, tutti gli altri vengono disabilitati mostrando una x rossa sopra di essi.

Quando l'utente ha completato la propria mossa, ovvero ha piazzato un familiare e ha eventualmente completato le azioni bonus e scelto i privilegi del consiglio, viene data automaticamente la possibilità di piazzare un familiare al giocatore successivo.

Quando tutti i giocatori non hanno più a disposizione familiari da piazzare (o anche se non hanno la possibilità di piazzarli) viene concluso il turno e iniziato quello successivo.

Alla fine dei turni pari viene effettuato il rapporto al vaticano, se un giocatore non ha punti fede a sufficienza per sostenere la chiesa, viene scomunicato automaticamente, e comparirà il cubo scomunica del suo colore nello spazio apposito sul tabellone. Se invece possiede abbastanza punti, viene mostrata una pop up chiedendo all'utente se ha intenzione o meno di sostenere la chiesa.

Al termine della partita viene mostrata la classifica dei giocatori ottenuta in base al calcolo dei punteggi.

In qualsiasi momento il giocatore ha la possibilità di abbandonare la partita premendo il bottone "Exit Game", verrà escluso dall'avanzamento delle mosse e gli altri giocatori continueranno a giocare normalmente.

Command Line Interface

Anche in questo caso l'utente ha la possibilità di iniziare una partita prima di raggiungere i 4 giocatori (sempre alla condizione di avere almeno un altro giocatore in attesa) tramite il comando 'iniziaPartita'.

Durante tutta la fase di gioco, ogni utente ha sempre a disposizione i seguenti comandi di default:

- vediRisorse: mostra un elenco delle risorse e punti a disposizione dell'utente
- vediCarte: mostra l'elenco delle carte nella plancia del giocatore
- vediDadi: mostra il valore dei dadi nel turno corrente di gioco
- vediCosto: dato il nome di una carta ne mostra il costo

Gli altri comandi sono limitati a determinate condizioni di gioco. All'inizio di ogni mossa viene mostrato il tabellone (in formato testuale) ai giocatori.

```
-------------------------------------------------------------------------------------------------------------------------------
| 4 CavaDiGhiaia               | | 8 Falegnameria              | | 12 Condottiero              | | 16 IngaggiareReclute        |
| 3 Bosco                      | | 7 Esattoria                 | | 11 Cavaliere                | | 15 CostruireLeMura          |
| 2 Borgo                      | | 6 Cappella                  | | 10 Badessa                  | | 14 CombattereLeEresie       |
| 1 AvampostoCommerciale       | | 5 ArcoDiTrionfo             | | 9 Artigiano                 | | 13 CampagnaMilitare         |
-------------------------------------------------------------------------------------------------------------------------------
17 - Spazio produzione piccolo     18 - Spazio produzione grande
19 - Spazio raccolto piccolo       20 - Spazio raccolto grande
21 - Spazio mercato 1              22 - Spazio mercato 2
23 - Spazio mercato 3              24 - Spazio mercato 4
25 - Spazio del Consiglio
```

Il giocatore che deve piazzare un familiare ha a disposizione solo (oltre a quelli di default) il comando 'piazza', che prevede i parametri del colore del familiare, il numero relativo allo spazio azione e il numero di servitori aggiunti per aumentare il valore dell'azione.

Nel momento in cui l'utente ha la possibilità di effettuare un'azione bonus o di scegliere un privilegio del consiglio vengono rispettivamente abilitati i comandi 'bonus' e 'privilegio'. Per il primo è necessario specificare parametri quali numero dello spazio azione e servitori aggiunti, per il secondo è sufficiente indicare l'indice del bonus (scelto tra quelli proposti a video).

Alla fine del periodo, se l'utente ha la possibilità di scegliere il sostegno alla chiesa, viene abilitato il comando 'sostegno' al quale bisogna semplicemente aggiungere il parametro 'si' oppure 'no'.

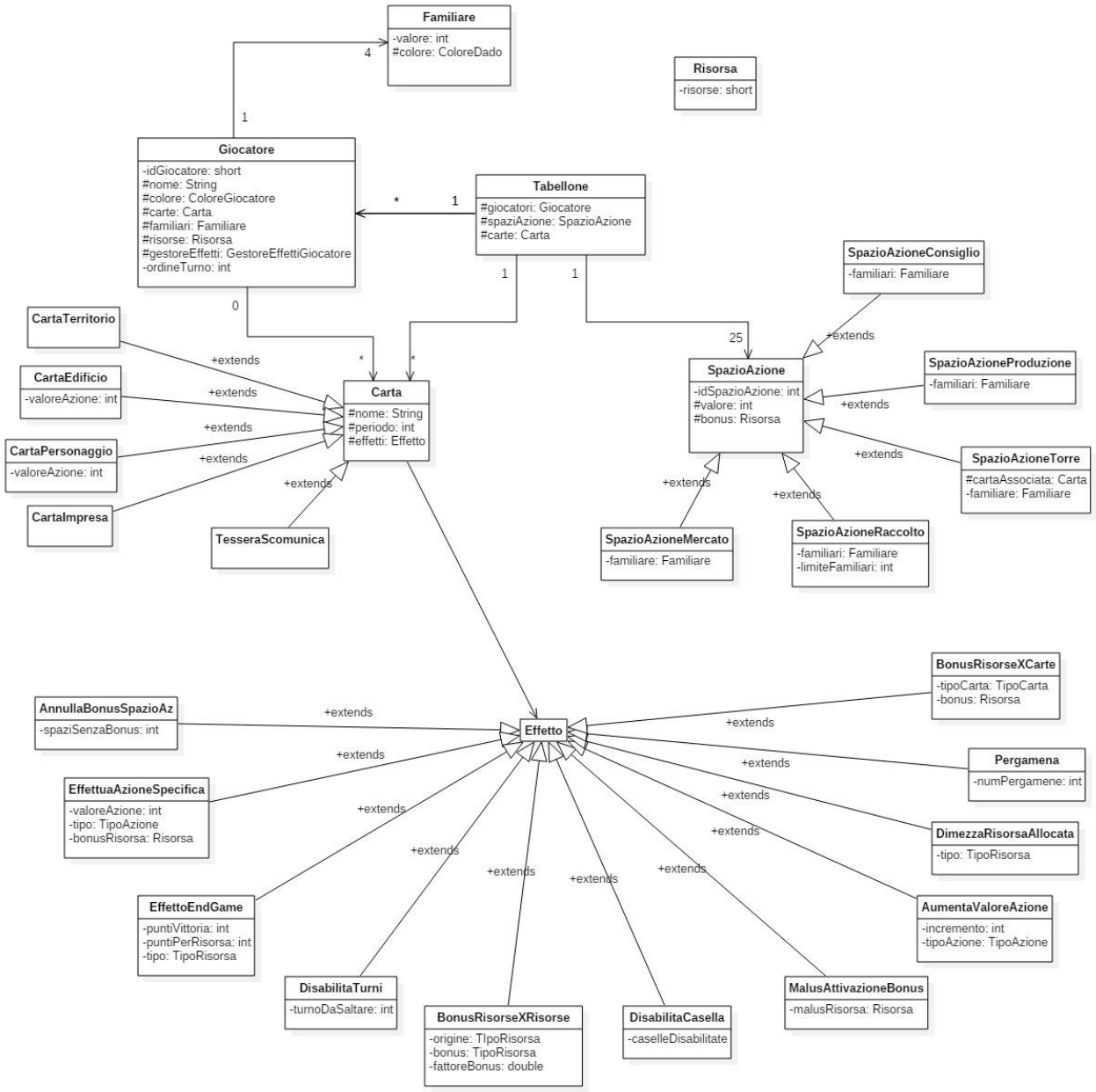
Il tabellone in formato testuale prevede che le carte acquistate dalla torre vengano colorate del colore del giocatore che le ha prese, in questo modo l'utente è conoscenza delle carte ancora disponibili e delle torri già occupate da altri giocatori.

Tutti gli errori di validazione ritornati dal server vengono mostrati a video colorati di rosso.

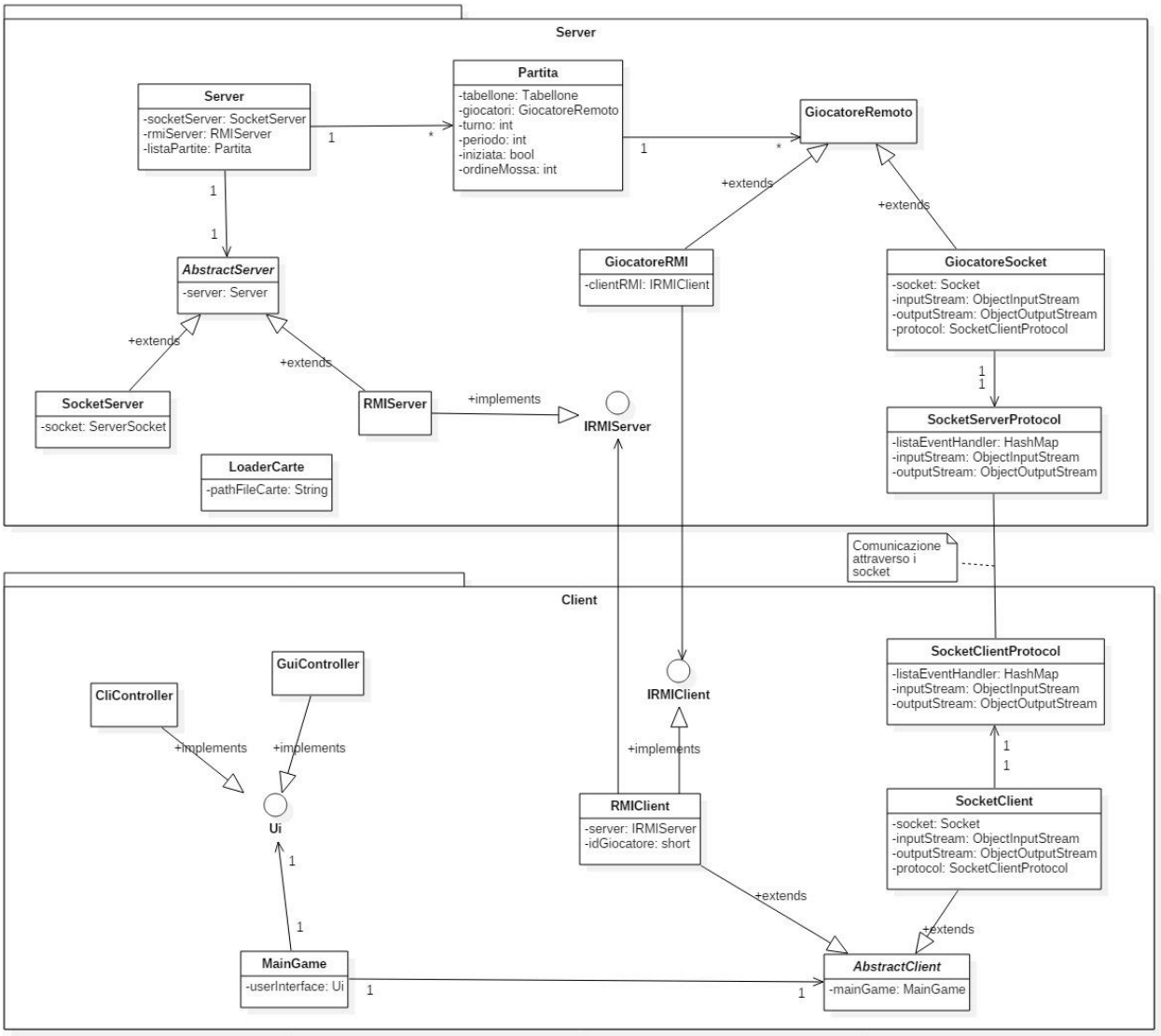
Tutte le volte che un utente prova a digitare un comando non corretto viene ritornato l'errore 'Comando sconosciuto', mentre viene specificato il caso in cui è stato sbagliato il numero di parametri attraverso l'errore 'Numero di parametri errato'.

Diagramma delle classi

Viene mostrato il diagramma delle classi relativo alle classi del package di Dominio.



Il prossimo diagramma delle classi è connesso al precedente, in quanto la classe Partita rappresenta il tramite per raggiungere la classe Tabellone. Il diagramma considera le classi nel package Server e i package del modulo Client (le classi relative all'interfaccia grafica vengono "collassate" in GuiController e CliController).

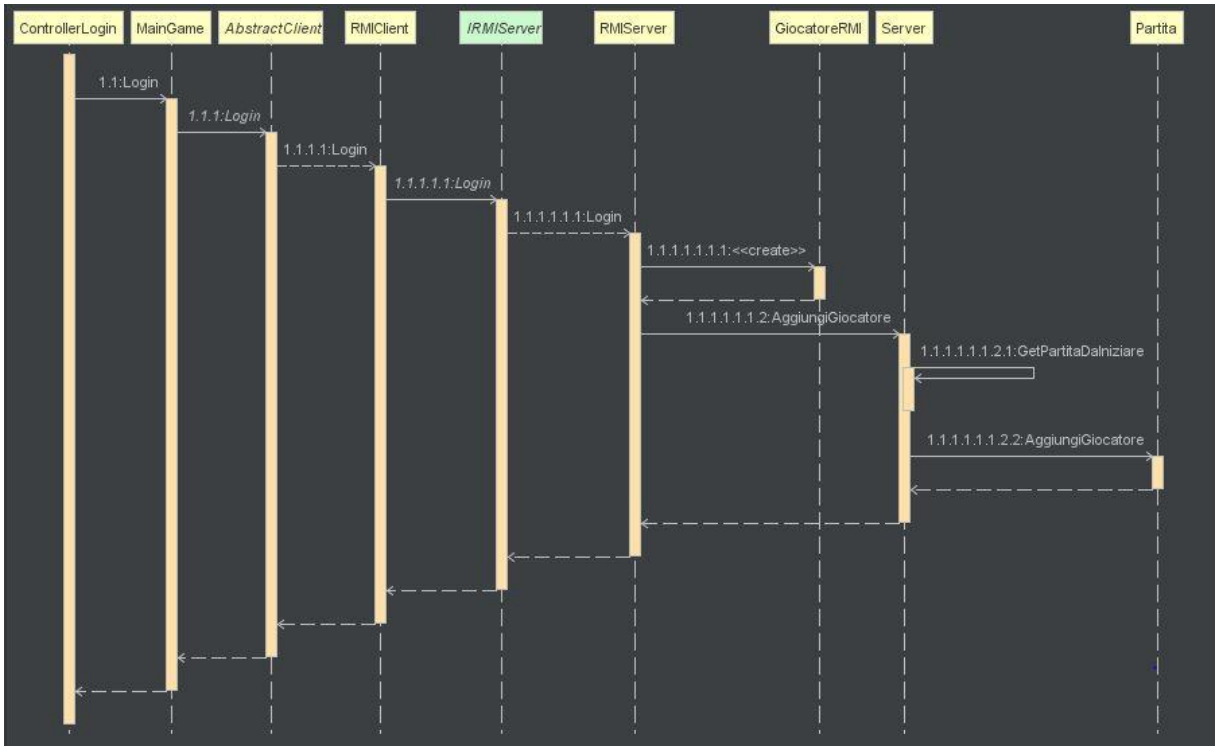


Sequence diagrams

Trattiamo il sequence diagrams di alcune delle funzioni principali del programma, ovvero la fase iniziale di login, una fase di gioco come ad esempio il piazzamento di un familiare, e la fase finale, nella quale vengono calcolati i punteggi dei giocatori e comunicati ai client. In questo modo si potranno vedere anche i diagrammi di tutti i metodi di comunicazione tra client e server. Per comodità, l'invio e la ricezione di messaggi tramite socket verranno divise in due esempi.

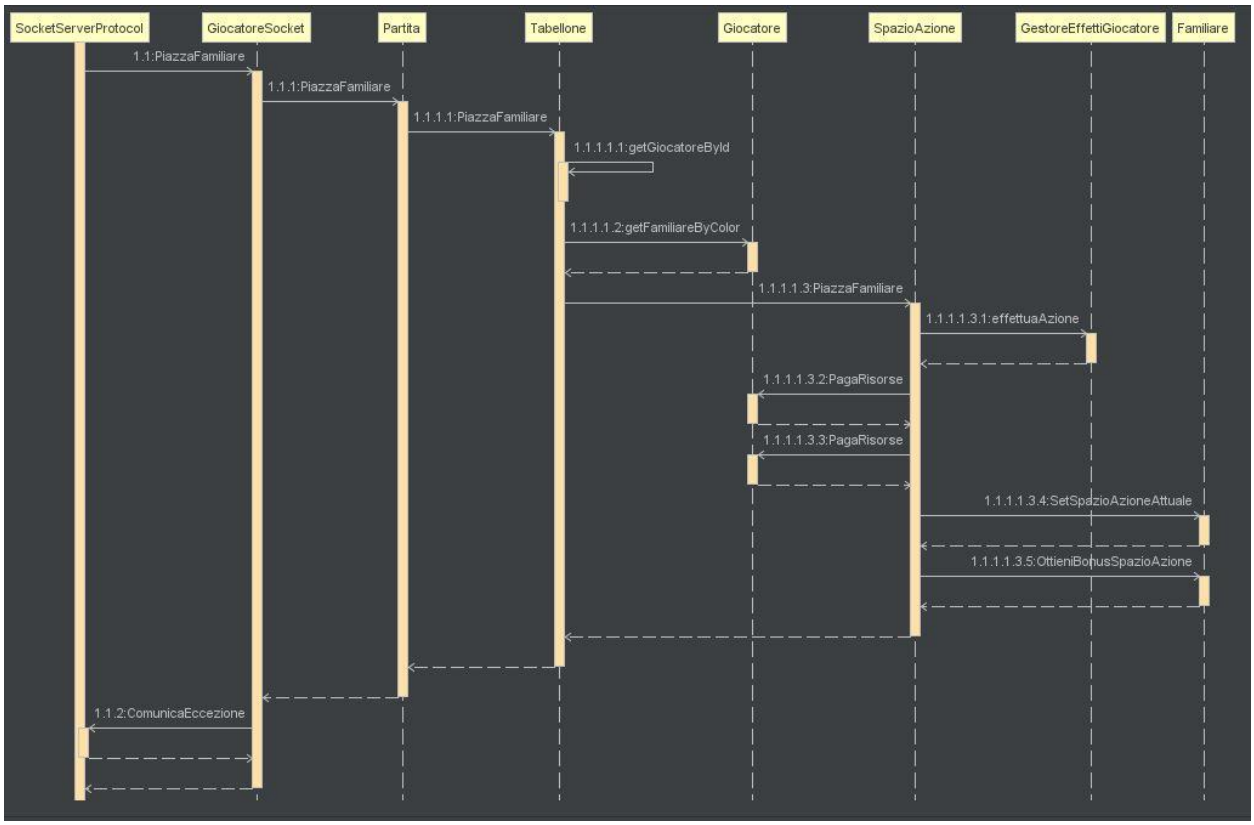
Fase di login

Partendo dal controller Login previsto dal pattern MVC, passando per il canale di comunicazione (abbiamo ipotizzato la scelta iniziale della tecnologia RMI), fino ad arrivare alla creazione del giocatore sul server e alla sua partecipazione alla partita.



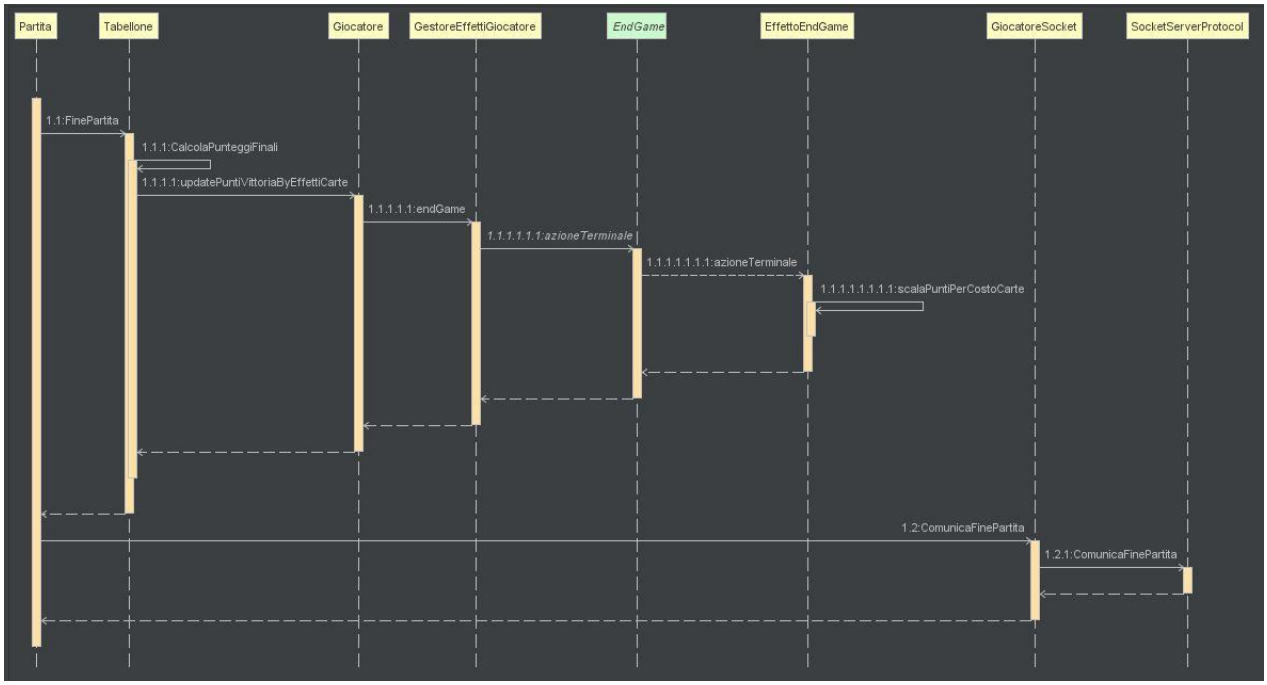
Piazzamento di un familiare

Viene mostrata la sequenza lato server, partendo dal ricevimento della chiamata da parte del client tramite socket.



Fine della partita

Calcolo dei punteggi lato server e comunicazione al client tramite Socket. Non include la ricezione del messaggio tramite socket in quanto già vista con l'esempio precedente lato server (duale a quello client).



Test

I test sono stati creati sfruttando il framework JUnit. È stata creata una classe di test per quasi tutte le classi di dominio presenti nel progetto, alcune di queste sono composte da metodi che compiono operazioni piuttosto banali, come ad esempio il set di alcune proprietà o semplici operazioni aritmetiche.

Anche questi metodi sono stati coperti dai test, ma per brevità verranno trattati solamente i test case più significativi. Per questi ultimi sono stati pensati dei metodi di Setup che inizializzano una situazione di gioco base, che poi viene alterata in ogni test nel modo più opportuno. Per ogni metodo delle classi di dominio dunque è previsto almeno un test case per verificarne il funzionamento nel caso in cui non vengano sollevate eccezioni, e anche un test case per ogni eccezione che può essere sollevata.

Le classi di test più significative si possono dividere in tre "sezioni":

Unit test sugli spazi azione

Nel metodo di setup inizializzano uno spazio azione del tipo specifico della classe di test (Torre, Produzione, Raccolto, Mercato, Consiglio).

Negli spazi azione viene controllato se il valore dell'azione è sufficiente per effettuare una determinata mossa, i test di validazione prevedono dunque un determinato valore dell'azione in ingresso, se la mossa non può essere effettuata il test si aspetta un'eccezione, in caso contrario deve terminare senza errori.

Lo unit test più corposo è rappresentato dallo SpazioAzioneTorreTest, che prevede molti più controlli. La fase di setup è la medesima degli altri spazi azione, ma i test prevedono anche la verifica di tutti gli effetti delle carte (comprese le tessere scomunica). Ogni test, dopo il Setup, prepara la sua specifica situazione di gioco, per esempio:

- Setta determinati valori alle risorse del giocatore
- Associa alla plancia di un giocatore qualche carta con determinati effetti permanenti
- Associa a uno spazio azione una determinata carta (con degli effetti immediati)

Dopo questa prima fase per la definizione dello scenario del test si procede con l'esecuzione del metodo da testare, in questo caso d'esempio si esegue il piazzamento del familiare nello spazio azione previsto.

Questo tipo di test prevede ovviamente la conoscenza del regolamento del gioco, infatti si andranno a verificare i vari attributi modificati in seguito al piazzamento del familiare, comprese le modifiche dovute agli effetti delle carte.

Sono previsti numerosi test di questo tipo, dovuti all'elevato numero di effetti delle carte da testare e alle diverse situazioni di gioco che si possono venire a creare.

Unit test sul Tabellone

Nel metodo di setup viene inizializzato un tabellone base, come quello che viene inizializzato all'inizio di una partita. I vari test case verificano le interazioni tra i giocatori, ogni test quindi recupera uno o più di essi. Oltre al già visto piazzamento di familiari, vi sono dunque metodi come il calcolo dei punteggi finali o il ricalcolo dell'ordine di gioco in funzione dei familiari piazzati nello spazio azione del consiglio, che prevedono appunto la verifica di tutti i giocatori.

Unit test sulla Partita

In questo unit test vengono fatti i vari controlli sull'andamento delle mosse e dei turni in funzione delle mosse effettuate dai giocatori, siccome l'avanzamento di mosse turni e periodi avviene in automatico.

Comunicazione client-server

La comunicazione tra client e server può avvenire sfruttando due diverse tecnologie: Socket e RMI.

Ogni client all'avvio può scegliere quale metodo di comunicazione utilizzare, in quanto il server è in grado di gestire entrambe le soluzioni contemporaneamente. Può accadere quindi che client che sfruttano i socket e client che invece adottano RMI partecipino alla stessa partita.

Server

All'avvio, il server configura entrambi i mezzi di comunicazione, inizializza il ServerSocket (che rimarrà in attesa di richieste di connessione da parte dei client tramite il metodo .accept()), ed effettua il binding del registro RMI, pubblicando l'oggetto remoto (Interfaccia implementata dal server RMI).

Il server socket adotta il seguente schema di funzionamento: per ogni client che si aggiunge, viene creato un thread dedito a gestirlo, questo thread è rappresentato dalla classe GiocatoreSocket, che si occupa di gestire la comunicazione solo con quel client attraverso un opportuno protocollo descritto in seguito.

Più semplice invece la gestione RMI, per ogni client viene istanziato un oggetto GiocatoreRMI, nel quale viene salvato il riferimento allo stub del client in questione. In questo modo tutte le comunicazioni verso il client RMI sfrutteranno questa classe.

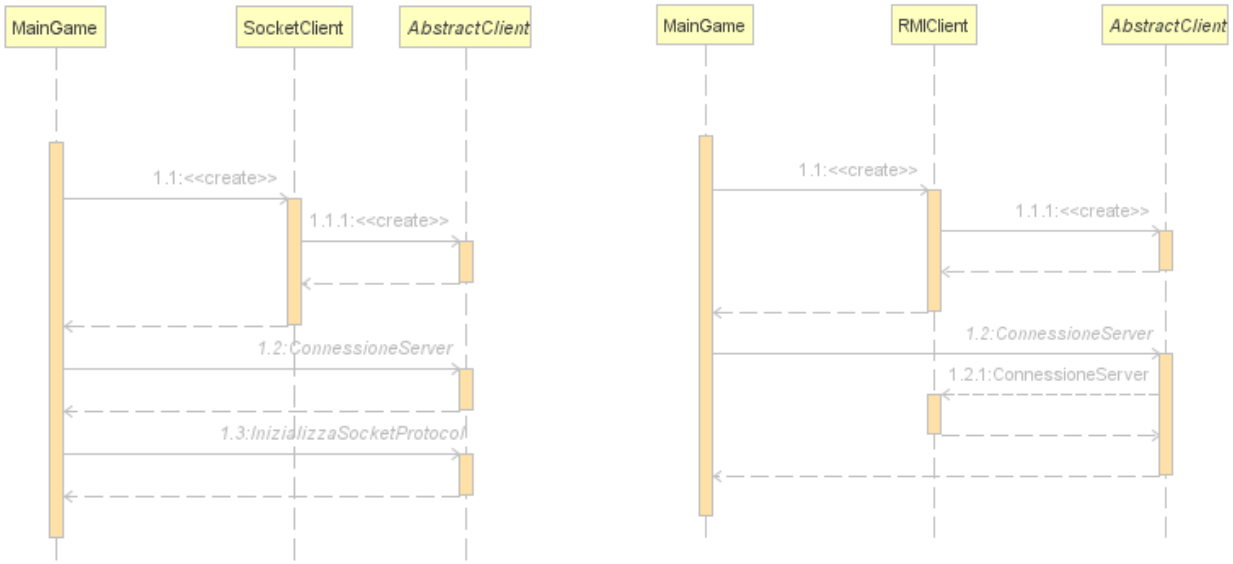
GiocatoreSocket e GiocatoreRMI estendono la classe GiocatoreRemoto, in questo modo gli oggetti che si occupano di mandare messaggi ai client possono implementare oggetti di tipo GiocatoreRemoto senza preoccuparsi del tipo di comunicazione scelta.

Client

Lato client invece verrà inizializzato solamente il canale di comunicazione scelto dall'utente.

Se è stato scelto di utilizzare RMI, viene istanziato un registro lato client, che verrà passato al server per permettere la comunicazione dal server verso il client (il server non dovrà eseguire lookup() alla ricerca dello stub caricato sul registro del client, in quanto gli verrà comunicato tramite la prima remote call), e ottiene un riferimento allo stub caricato dal server.

Se invece l'utente ha preferito i socket, viene creata una istanza della classe Socket e vengono inizializzati gli stream di comunicazione in input e output (ObjectInputStream e ObjectOutputStream). Inoltre, la coesistenza tra i due canali di comunicazione (dovendo gestire anche RMI) rende necessario l'utilizzo di un protocollo di comunicazione tra i socket.



Infatti, per rendere trasparente questa scelta al livello 'presentation', le classi RMIClient e SocketClient estendono la classe astratta AbstractClient, la quale rappresenta il modello generico di comunicazione verso il server. Tuttavia, le due tecnologie necessitano di diversi approcci per la comunicazione, in particolare ai socket è spesso richiesto un maggior numero di chiamate per avere lo stesso risultato ottenibile attraverso un'invocazione remota di metodi. Per questo motivo è stato implementato il sopracitato protocollo di comunicazione, che regola l'invio e la ricezione di informazioni tra client e server.

In quanto protocollo viene implementato sia dal client che dal server, seguendo lo stesso principio di funzionamento, per simulare una invocazione remota di metodo:

- Viene effettuata una prima scrittura sullo stream per rendere noto al ricevente il metodo da implementare.
- Il ricevente attraverso un apposito handler esegue il metodo adatto alla ricezione dei messaggi successivi previsti dal protocollo, e resta in attesa dei parametri in input.
- A questo punto il chiamante scrive in sequenza tutti i parametri necessari all'esecuzione del metodo, che vengono letti uno ad uno dal chiamato.
- Terminato l'invio e la lettura dei parametri, il chiamato ha a disposizione tutte le informazioni necessarie per eseguire le operazioni richieste dal chiamante (come se fosse stata eseguita un'invocazione remota)

Segue un esempio di come avviene la comunicazione fino alla classe 'Partita' in seguito ad un evento a livello di interfaccia lato client: